Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ПО ДИСЦИПЛИНЕ:

ТИПЫ И СТРУКТУРЫ ДАННЫХ

ТЕМА: РАБОТА СО СТЕКОМ

Студент **Поздышев А. В.**

Группа **ИУ7-31Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____**Поздышев А. В.**

Преподаватель _____**Барышникова М. Ю.**

2024

ЦЕЛЬ РАБОТЫ

Реализовать операции работы со стеком, который представлен в виде динамического массива и в виде односвязного линейного списка; оценить преимущества и недостатки каждой реализации; получить представление о механизмах выделения и освобождения памяти при работе со стеком.

УСЛОВИЕ ЗАДАЧИ

Перевести выражение в постфиксную форму с учетом приоритета выполнения операций.

Исходные данные

Стек в виде динамического массива представлен как массив содержащий тип char и указатель на последний элемент массива (головы).

Стек в виде связанного списка представлен как структура, содержащая значение char и указатель на предыдущий элемент списка.

Способ обращения к программе

Пользователь через консоль запускает программу: ./app.exe. После чего программа выводит меню опций.

Меню программы:

1. Вывести стек массива.
2. Вывести стек списка.
3. Добавить элемент в стек массива.
4. Добавить элемент в стек списка.
5. Получить обратную польскую запись со стеком массива.
6. Получить обратную польскую запись со стеком списка.
7. Получить историю стека для польской записи со стеком массива.
8. Получить историю стека для польской записи со стеком массива.
9. Удалить элемент из стека массива.
10. Удалить элемент из стека списка.
11. Замерить время добавления и удаления n элементов для стеков.
12. Вывести результаты времени выполнения обратной польской записи.
0. Выход.

Стеки выводятся с головы до конца, то есть голова является первым элементом строки вывода. Для операций 5 и 6 пользователю следует ввести

арифметическое выражение, содержащее только числа и операции “+-*/^ ()” в инфиксном виде.

Описание возможных аварийных ситуаций и ошибок пользователя

Аварийные ситуации:

- Некорректный ввод: Арифметическое выражение содержит символы;
- Некорректный ввод: Арифметическое выражение не в инфиксном виде;
- Некорректный ввод: Арифметическое выражение содержит неверное расположение скобок “)(“;
- Переполнение: В стек массива добавляется элемент, не влезающий в массив;
- Опустошение: Попытка удаления элемента из пустого стека.

ОПИСАНИЕ ВНУТРЕННИХ СТРУКТУР ДАННЫХ

Стек массив представлен с помощью структуры stack_t

```
typedef struct stack_massive
{
    char *massive;

    char *phead;

    size_t max_len;

} stack_t;
```

massive – указатель на начало массива типа char;

phead – указатель на последний элемент массива;

max_len – размер выделенного массива.

Стек список представлен с помощью двух структур list_t и node_t.

```
typedef struct node
```

```
{
    struct node *prev;
    char symbol;
} node_t;


typedef struct list
{
    node_t *phead;
} list_t;
```

node_t – элемент списка (узел).

struct node *prev – указатель на предыдущий элемент списка;

char symbol – значение элемента.

list_t – список элементов

node_t *phead – указатель на последний элемент стека.

ФУНКЦИИ ПРОГРАММЫ

Init_stack

Заголовок:

```
int init_stack(stack_t *stack);
```

Функция инициализирует стек массив. Принимает указатель на структуру stack_t. Возвращает код ошибки.

Init_stack

Заголовок:

```
int init_list(list_t **list);
```

Функция инициализирует стек список. Принимает двойной указатель на структуру list_t. Возвращает код ошибки.

Free_list

Заголовок:

```
void free_list(list_t *list);
```

Функция освобождает память под список. Принимает указатель на структуру list_t. Возвращает код ошибки.

Add_tostack

Заголовок:

```
int add_tostack(stack_t *stack, char symbol);
```

Функция добавляет элемент в стек массива. Принимает указатель на структуру stack_t, добавляемый символ symbol. Возвращает код ошибки.

Pop_frmstack

Заголовок:

```
int pop_frmstack(stack_t *stack);
```

Функция удаляет последний элемент из стека массива. Принимает указатель на структуру stack_t. Возвращает код ошибки.

Take_frmstack

Заголовок:

```
int take_frmstack(stack_t *stack, char *taken);
```

Функция забирает последний элемент из стека массива. Принимает указатель на структуру stack_t и указатель на получаемый символ taken. Возвращает код ошибки.

Add_tolist

Заголовок:

```
int add_tolist(list_t *head, char const symbol);
```

Функция добавляет элемент в стек списка. Принимает указатель на структуру list_t, добавляемый символ symbol. Возвращает код ошибки.

Pop_frmlist

Заголовок:

```
int pop_frmlist(list_t *head);
```

Функция удаляет последний элемент из стека списка. Принимает указатель на структуру list_t. Возвращает код ошибки.

Take_frmlist

Заголовок:

```
int take_frmlist(list_t *head, char *taken);
```

Функция забирает последний элемент из стека списка. Принимает указатель на структуру list_t и указатель на получаемый символ taken. Возвращает код ошибки.

Print_stack

Заголовок:

```
void print_stack(stack_t *stack);
```

Функция выводит стек массива слева направо. Принимает указатель на структуру stack_t.

Print_list

Заголовок:

```
void print_list(list_t *head);
```

Функция выводит стек списка слева направо. Принимает указатель на структуру stack_t.

Convert_topostfix_stack

Заголовок:

```
int convert_topostfix_stack(char *postfix, char *expression, stack_t stack_store);
```

Функция переводит арифметическое выражение в постфиксную форму с помощью стека массива. Принимает указатели на строки postfix, expression, stack_store содержащий историю стека массива. Возвращает код ошибки.

Convert_topostfix_list

Заголовок:

```
int convert_topostfix_list(char *postfix, char *expression, list_t *list_store);
```

Функция переводит арифметическое выражение в постфиксную форму с помощью стека списка. Принимает указатели на строки postfix, expression, list_store содержащий историю стека списка. Возвращает код ошибки.

ОПИСАНИЕ АЛГОРИТМА

Алгоритм выводит меню действий и ожидает ввод пользователя.

0. Действие 1:
    a. Вывод стека массива.
1. Действие 2:
    a. Вывод стека списка.
2. Действие 3:
    a. Добавить элемент в стек массива.
    b. Проверка успешности операции.
3. Действие 4:
    a. Добавить элемент в стек списка.
    b. Проверка успешности операции.
4. Действие 5:
    a. Ввод арифметического выражения.
    b. Проверка арифметического выражения.
    c. Конвертация в постфиксный вид с помощью стека массива.
    d. Сохранения замера выполнения операции.
5. Действие 6:
    a. Ввод арифметического выражения.
    b. Проверка арифметического выражения.
    c. Конвертация в постфиксный вид с помощью стека списка.
    d. Сохранения замера выполнения операции.
6. Действие 7:
    a. Вывод истории стека массива.
7. Действие 8:
    a. Вывод истории стека списка.
8. Действие 9:
    a. Удаление элемента их стека массива.
    b. Проверка успешности операции.
9. Действие 10:
    a. Удаление элемента их стека списка.
    b. Проверка успешности операции.
10. Действие 11:
    a. Ввод кол-ва добавляемых элементов в стеки.
    b. Вычисление времени выполнения добавления в стеки.
    c. Вычисление времени выполнения удаления их стеков.

d. Вывод результатов.

11. Действие 12:
    a. Проверка на наличие замеров.
    b. Вывод замеров выполнений обратной польской записи.
12. Действие 0:
    a. Выход из программы

НАБОР ТЕСТОВ

Тест	Входные данные	Выходные данные
Некорректный ввод: Ариф. Выражение содержит символы	1+1-a+1	ERR_IO: Неверная арифметическая запись.
Некорректный ввод: Ариф. Выражение не в постфиксном виде	1++1-2	ERR_IO: Неверная арифметическая запись.
Некорректный ввод: Ариф. Выражение содержит неверное расположение скобок	1+1-)(1+1	ERR_IO: Неверная арифметическая запись.
Опустошение: удаление элемента из пустого стека списка	<пустой стек список>	ERR_EMPTY: Стек пуст.
Опустошение: удаление элемента из пустого стека массива	<пустой стек массив>	ERR_EMPTY: Стек пуст.
Переполнение: добавление элемента не влезающего в стек массива.	1	ERR_OVERFLLOW: Переполнение стека

МЕТОДИКА ЗАМЕРОВ

Вычисление замеров добавления или удаления n элементов проводилось следующим образом: для каждого стека находилось среднее за 10 замеров.

ЗАМЕРЫ

N, кол-во элементов	Удаление из стека массива, мс	Добавление в стек массива, мс	Удаление из стека списка, мс	Добавление в стек списка, мс
100	0.002	0.006	0.007	0.008
500	0.001	0.013	0.02	0.024
800	0.002	0.017	0.026	0.03
1000	0.001	0.018	0.03	0.045

N, длина арифметического выражения	Время выполнения операции со стеком массива, мс.	Время выполнения операции со стеком списка, мс.
100	0.009	0.015
200	0.011	0.020
500	0.014	0.021
1000	0.015	0.025

Можно сделать вывод, что работа операций добавления и удаления со стеком массива в среднем эффективней в 1.8 раз.

Работа стека массива с постфиксной записью в среднем в 1.6 раз быстрее чем со стеком списка.

ПАМЯТЬ

N, кол-во элементов	Объем памяти для стека массива, байт.	Объем памяти для стека списка, байт.
100	1024	1608
200	1024	3208

500	1024	8008
1000	1024	16008

Во время выполнения замеров мною было выяснено, что sizeof(node_t) = 16 байт и что каждый элемент списка расположен друг за другом по адресу кратному размеру структуры. Адрес конца стека принимает наибольший адрес. При удалении и повторном добавлении элемента он располагается в той же области памяти. Компилятор выравнивает список и эффективно располагает его. Можно сделать вывод, что фрагментация памяти отсутствует.

При добавлении элемента в стек заполненного массива, в случае если размер массива не достиг максимального, то выделенный блок памяти пере выделяется под в два раза увеличенный, иначе программа прекращает работу с кодом ошибки переполнения.

ВЫВОД

Наиболее эффективным по времени выполнений операций удаления и добавления является стек массива. В среднем первый способ в 1.8 раз быстрее за счет того, что ему нет необходимости выделять новые блоки памяти под добавленные элементы. Главным минусом стека массива является выделение неиспользованной памяти (для динамического массива нужно пере выделять целый блок при переполнении). В случае списка задействуется вся необходимая память.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Что такое стек?

> Стек — это структура данных, работающая по принципу "последний пришёл — первый вышел". Элементы добавляются и удаляются только с одного конца, называемого вершиной стека.

2. Как и сколько памяти выделяется под хранение стека при различной его реализации?

Выделяют два вида реализации:

Через статический (динамический) массив: память выделяется статически (фиксированное количество элементов) или динамически (например, с помощью malloc). Объём памяти определяется размером массива.

Через связный список: память выделяется динамически под каждый узел (элемент) при добавлении, и она зависит от количества элементов в стеке и размера хранимых данных.

3. Как освобождается память при удалении элемента стека при различной реализации?

Массивная реализация: при удалении элемента просто уменьшается указатель вершины, фактически без освобождения памяти.

Связный список: память освобождается с помощью free, чтобы удалить узел.

4.Что происходит с элементами стека при его просмотре?

При элементы не удаляются: просто возвращается значение верхнего элемента стека, а структура остаётся неизменной.\

5. Как эффективнее реализовывать стек? От чего это зависит?

Эффективность реализации стека зависит от:

Для неизвестного числа элементов: предпочтительным будет связный список — он гибок по размеру.

Для работы с определенным кол-вом данных или фиксированным: лучше всего может подойти массивная реализация, которая быстрее по доступу.

В моем случае динамические массивы позволяют лучше управлять памятью без фрагментации, но могут быть медленнее из-за затрат на пере выделение и освобождение памяти.